Method to Select the Most Suitable Software Tool for the Development of an Hmi Virtual Prototype

Luca Minin, Roberto Montanari, Stefano Marzani, Francesco Tesauri, and Luca Canovi

University of Modena and Reggio Emilia, Department of Science and Methods of Engineering, Via Amendola 2, Padiglione Tamburini, 42100 Reggio Emilia, Italy
{canovi.luca, marzani.stefano, minin.luca, montanari.roberto, tesauri.francesco}@unimore.it

Abstract. This paper proposes a method which allows user interfaces practitioners to choose the most suitable software tool for the development of a Human Machine Interface (HMI) virtual prototype. In the design process of a User Interface (UI), the prototyping activity represents an important way to detect design errors since the beginning of the development process, saving time and money. The choice has to take care of the technical features, designers' requirements and prototyping characteristics of the tool. The methodology includes four steps: 1) research on the most diffused virtual prototyping software employed in the prototyping activity; 2) benchmark on the technical and virtual prototyping features of each selected tool; 3) creation of a method to asses and classify each prototyping tool considering their specific characteristics; 4) application of the method to a case study.

Keywords: user centered design, virtual prototyping, user interface, HMI.

1 Introduction

In parallel with the wide spreading of the User-Centred Design approach (UCD – Norman, 1998), prototyping has become the key option to actually "user-centre" a User Interface (UI).

There are many perspectives from which the benefits of the prototyping can be outlined. From a financial point of view, for example, a long tradition of studies have revealed how cost-valuable prototyping activities are (Winters et al., 2004; Hellestrand, 1999; NASSDA, 2003); in fact, the virtual prototyping activity allows designers to correct design errors in the early stage of the process, in order to reduce the overall re-design costs.

Several prototyping tools (mainly software tools) can be exploited in all phases of the design process, from early stages to the engineering one (Dix, 2004).

To easily find design errors, it is necessary to choose the most suitable virtual prototyping software which fits not only with the requirements of the end user but

D. Harris (Ed.): Engin. Psychol. and Cog. Ergonomics, HCII 2007, LNAI 4562, pp. 118–127, 2007.

also with the final employment of the virtual model. For instance, whether the virtual prototype model will be thrown away after an usability or functionality test or if it will be used as a basis for the development of a part of the final system.

A virtual prototype may be classified (Dix, 2004; Norman, 1998) according to: a) the prototyping technique used for designing the virtual model, b) the fidelity of the prototype in relation to the real system, c) the number of functions implemented by the prototype and d) the structure of the tasks.

A benchmark analysis among virtual prototyping tools was carried out in order to assess and classify them according to their technical features, designer's needs and their potential to create a virtual prototype with specific characteristics.

This classification allowed to create a method for evaluating whether a specific prototyping tool may be used for developing a prototype starting from the requirement of the users and the designers. Finally, this method was applied to a use case experiment.

2 Virtual Prototyping Tools' Sample

A benchmarking was conducted in 2005 (Minin, 2005) on academic and industrial research groups (N = 46) involved in different domains of Human Factors (HF) studies (automotive, aeronautic, nomadic devices, medical, etc.) in the US and Europe. The analysis paid particular attention to the virtual prototyping tools adopted by each group; a selection of the most widespread tools was made in order to carry out an analysis of their technical and virtual prototyping features.

The selection of the most widespread virtual prototyping tools is listed below:

- Altia Design 5.28;
- Macromedia Director Mx 2004;
- Rapid Plus 8.0;
- Gaio Protobuilder;
- MS Power Point 2003.

Subsequently, a benchmark analysis of the selected software was developed focusing on the following three macro-characteristics: (1) designer needs (2) technical features and (3) the potential of the software to create a virtual prototype of a user interface with specific characteristics.

The analysis was carried out by a group of four experts of the University of Modena and Reggio Emilia (Italy) composed by two virtual prototyping designers and two Human Factors (HF) experts.

3 Benchmark Analysis

As mentioned above, a benchmark analysis among the selected prototyping tools was carried out. Starting from previous works (Szekely, 1995; Myers, 1995; Righetti, 2006) the team of designers and HF experts selected several features of the tools

which affect the development of a prototype. These features were classified into three groups:

1. designers' needs;
2. technical features useful for virtual prototyping activity;
3. features related to the ability of the tool to create a virtual prototype with specific characteristics.

In the following paragraphs a description of the three different groups of features among the prototyping tools is depicted. In order to compare the prototyping tools, firstly a method for the evaluation of the features is presented.

Finally, the tables regarding the benchmark analysis are presented.

3.1 Method for the Evaluation of the Features

Each feature which belongs to the first and second group above mentioned is described by only one of the following characteristics:

- *Availability/Accessibility*: features could be totally available, partially available or absent in the prototyping tool. For instance, not all the tools allow to create automatic data logs which record the interaction between the prototype and the final user;
- *Complexity*: for available/accessible features is specified whether it is very complex, on the average complex or simple to be used. For instance, some prototyping tools allow to program the logic of the prototype only through a low-level programming language (complex to be used).

The degree of Availability/Accessibility and Complexity defined for each feature are based on a 3-level color scale, shown in the following table.

Table 1. Three level color scale

Colour	Availability/Accessibility	Complexity
Green	The feature is totally available in the prototyping tool.	The complexity of use of the feature is low.
Yellow	The feature is partially available in the prototyping tool.	The complexity of use of the feature is on the average.
Red	The feature is not available in the prototyping tool.	The complexity of use of the feature is high.

3.2 Designers' Needs

In the following table, the most relevant designers' needs and the related characteristics (in brackets) are numbered and described.

Table 2. Designers' needs

Number	Description of the feature
0	*Computer knowledge* (complexity). Knowledge needed for the prototype development: graphic editing (green), graphic and high-level programming (yellow) or graphic and low-level programming (red).
1	*Programming language* (complexity). The logics of the prototype can be developed: only through a low-level programming language (red), high level (green) or the tool offers both programming levels (yellow).
2	*Complexity and accuracy of the tutorial* (complexity). The use of the tutorial is: complex (red), on the average complex (yellow) or easy (green).
3	*Availability of a tool for development of the graphic interface* (Availability/Accessibility). A tool for the development of all the prototype's graphic interface is: available inside the prototyping software (green), only simple graphics can be created (yellow) or all the graphics have to be created with an external tool (red).

3.3 Technical Features Useful for Virtual Prototyping Activity

In the following table, the most relevant technical features and the related characteristics (in brackets) are numbered and described.

Table 3. Technical features

Number	Description of the feature
4	*Generation of a stand-alone application* (Availability/Accessibility). The tool allows to create a stand alone application of the prototype which can be loaded in a computer: without licence (green), only with licence (yellow) or a stand alone application can not be created (red).
5	*Generation of programming code* (Availability/Accessibility).The tool allows to generate code of the prototype: in different programming languages (green), in only one language (yellow) or no code generation is possible (red).
6	*Connection to external applications* (Availability/Accessibility). The tool allows to connect the prototype to other applications able to manage: both the graphic and the logic through specific API (green), either of the two (yellow) or the connection is not available (red).
7	*Library of graphical objects with already implemented programming code* (Availability/Accessibility). This kind of library is available and the programming code of the objects is editable (green), not editable (yellow) or the library is not available (red).
8	*Data log for usability and functionality tests* (Availability/Accessibility). An automatic log of the interaction between the user and the prototype is provided (green), only a manual log is provided – that is, the log has to be configured using programming languages (yellow), or it is not possible to generate a data log (red).

Table 3. (*continued*)

9	*Generation of specification documents* (Availability/Accessibility). Specification documents related to the functionality and the components of the prototype with an high detail can be generated (green), can be generated with a low detail – that is, only the components but not the functionality (yellow) or specification documents can not be generated (red).

3.4 Features Related to the Potential of the Software to Create a Virtual Prototype for a Specific Kind of Use Employment

In this section, the features of the tools which allow the designer to create prototypes with specific characteristics are described. These characteristics, selected from the literature (Mayhew, 1992; Sommerville, 1995; Dix, 2004; Norman, 1998) are the following:

A. the prototyping technique (Throw-away, Incremental or Evolutive);
B. the fidelity (High, Medium or Low);
C. horizontal or vertical prototypes;
D. the structure of the tasks.

In the following paragraph each point of the list is described. The team of designers and HF experts defined a correlation between specific designers' needs and technical features with the prototyping characteristics listed above. In the following paragraph, each feature described in Table2 and Table3. Table4 call back the related number (e.g. feature 4, feature 6, etc.).

3.4.1 Prototyping Technique

The objective of the *throw-away prototyping* is to understand the system requirements. Starting from a stand alone application of the prototype, iterative tests with the end users on the usability and functionality of the final system are carried out. Finally, the specifications of the final system are defined. Thus, the prototyping tool should provide features 4 and 8 for the testing activity and feature 9 for the development of the system's specifications.

Incremental prototyping aims at developing a virtual prototype for each part of a modular system (i.e. a system composed by several components). Starting from an overall architecture of the final system, each prototype is created and tested with users (features 4 and 8 required); then it is delivered in increments with the related specification documents (feature 9) and the programming code (feature 5) is used for the development of a part of the final system.

The objective of *evolutionary prototyping* is to deliver a working system to end-users. The technique is similar to the incremental but with an evolving design of the overall system. Then, the same features of the incremental prototyping should be provided. However, the specification documents and the generated programming code are related to the overall system and are delivered at the end of the iterative design activity.

The following table is a summary of the technical features required for the development of the three prototyping techniques. The numbers of the columns

represent the technical features listed in Table2, Table3 and Table4. As may be seen in Table1, "G" and "Y" correspond to Green and Yellow, the colours which identify that a specific technical feature is available in the prototyping tool. An empty cell means that the technique does not require the specific technical feature.

Table 4. Prototyping techniques (rows) and required technical features (columns)

	4	5	8	9
Throw away	G,Y		G,Y	G,Y
Incremental	G,Y	G,Y	G,Y	G,Y
Evolutive	G,Y	G,Y	G,Y	G,Y

3.4.2 Fidelity of the Prototype

Regarding the fidelity of the prototype, for each level (High, Medium, Low), the prototyping tool should provide the following features:

- *High fidelity*: the prototype should be similar to the final system and it should work in the same way. Feature 4, 5, 6 are required. The tool should have all the features required to develop a prototype which can also be used as a basis of the final system.
- *Medium Fidelity*: useful for testing design concepts early in the design process. Features 4, 6 are required. The prototype should look like the final system, but it is not required it will also be the basis for the development of a part of it, then the prototype does not provide all the interactions and functions of the final system.
- *Low fidelity*: it represents a mock-up of a system (or part of a it) used to support user studies. Feature 4 is required. The prototype should be developed in a rapid way. It represents only a set of functions or interactions of the final system.

As for Table 4 Table 5, the technical features of the prototyping tool required for the development of a High, Medium or Low-Fi prototype are depicted below.

Table 5. Level of fidelity (rows) and required technical features (columns)

	4	5	6
High-Fi	G,Y	G,Y	G,Y
Medium-Fi	G,Y		G,Y
Low-Fi	G,Y		

3.4.3 Horizontal/ Vertical Virtual Prototypes

An horizontal prototype displays "breadth" of functionality, that is, broad but only top-level functions of the final system. In a vertical prototype full functionality and performance of a small part of the final system are implemented. Starting from this definition, the ability of a prototyping tool to allow the development of prototypes with these characteristics was estimated. For instance, Altia Design has a set of applications which allows to easily develop multilayered virtual prototypes although the levels may only be displayed and managed by the designer one by one and additionally, the workspace is quite limited in width (i.e. it is difficult to manage a wide range on functions characterized by different level of deepness). Hence, this tool is a good solution for the development of a deep vertical prototype but not an horizontal one.

3.4.4 Structure of the Tasks

When the user has to complete a specific task, he/she has to perform a sequence of interactions with or without the help coming from the system (e.g. affordances and feedbacks). Alternatively, a part of the task can be automated (less interactions). The more the task is automated, the easier the structure of the task will be. The ability of the tool to allow the designer to develop prototypes with an easy structure of the tasks is evaluated. If the level of the programming language for the development of the prototype's logic is medium-high and a library of graphical objects with already implemented programming code is available in the tool (point 1 and 7 of Table2, Table3), then a more simple structure of the task can be implemented easily since some tasks may be automated rapidly.

Table 6. Structure of the tasks (rows) and technical features/designers' needs (columns)

	1	**7**
Easy structure of the task	G,Y	G,Y

3.5 Benchmark Among the Prototyping Tools

Finally, all the above mentioned features were analyzed among the selected tools in order to assess the overall characteristics. Three tables for the benchmark analysis were prepared:

- the first including the comparison of the features related to the designers' needs Fig.1;
- the second including the technical features Fig.2;
- the third including the features related to the ability to create prototypes with specific characteristics Fig.3.

The last table is the result of the combination of the data contained in the first two tables, as described in the previous paragraph. A "X" symbol in the table means that the feature is available, otherwise the cell is empty (not available).

A part of the three tables is depicted in the following figures; in the original tables all the features of the previous paragraph were reported with a small description.

	Altia Design	Macromedia Director Mx 2004	Rapid Plus 8.0	Protobuilder	MS Power Point 2003
Computer knowledge	R	Y	R	Y	G
Programming language	Y	G	Y	G	G
Complexity and accuracy of the tutorial	G	Y	G	G	Y

Fig. 1. Benchmark on the designers' needs

	Altia Design	Macromedia Director Mx 2004	Rapid Plus 8.0	Protobuilder	MS Power Point 2003
Generation of a stand-alone application	G	G	G	Y	R
Generation of programming code	G	Y	G	Y	R
Connection to external applications	G	Y	Y	R	R

Fig. 2. Benchmark on the technical features

	Altia Design	Macromedia Director Mx 2004	Rapid Plus 8.0	Protobuilder	MS Power Point 2003
Throw away	X	X	X	X	X
Incremental	X	X	X	X	
Evolutive	X	X	X	X	
High fidelity	X				
Medium fidelity	X	X	X	X	

Fig. 3. Benchmark on the prototyping characteristics

4 Method to Asses and Classify Each Prototyping Tool Considering Their Specific Characteristics

The method to select the best prototyping tool for the development of a HMI prototype follows the steps:

1. definition of the characteristics of the prototype (see § 3.4);
2. definition of the skills and the requirements of the team which design the prototype (see § 3.2);

3. definition of the technical features required for the creation of the prototype (see § 3.3);
4. selection of the prototyping tool which fits with the characteristics defined in steps 1 to 3 (see § 3.5).

Each step of the list is described in the previous paragraphs. At step 4, the method was applied to a selection of prototyping tools but it can also be enlarged to other software. An application of the method is described in the following paragraph.

4.1 Use Case Application

The above mentioned method was applied to the development of a virtual prototype of an in-vehicle display.

The designers and the project owner required that a High-Fi and vertical prototype had to be created through an Evolutive technique. The team of designers was composed by three computer scientists and one HF expert. It was required to develop at first a stand alone application of the system for usability tests. Then, after a re-design of the system based on the results of the test, the owner required to connect the prototype to the vehicle network and test the functionalities of the prototype in the real use environment.

By comparing these requirements with the features of the prototyping tools depicted in the figure of § 3.5, the selected tool was Altia Design. In fact, it would allow to create virtual prototypes with the required characteristics. Moreover, it is possible to connect Altia to other software, in particular Matlab Simulink, and to create an interface with the vehicle network. After that, a stand alone application and data-log for usability tests can be provided. Finally, using Matlab Simulink and Deepscreen (i.e. the code generator tool of Altia Design), the programming code of the prototype can be generated and used as a basis for the development of the real in-vehicle interface.

5 Conclusions

This paper proposes a solution to help designers in choosing the most fitting software tool for the development of a virtual prototype with specific characteristics. Analyzing the technical features and the designers' requirements with certain degrees of availability, accessibility and/or complexity, it is possible to assess whether a prototyping tool is able to create a horizontal/vertical prototype with a high/medium/low-fidelity or an easy structure of the task. Moreover, it is possible to define which prototyping techniques the specific tool allows to use.

Since there are a lot of different prototyping tools with different characteristics, it is important to find the most suitable solution which fits with the expectations of the designer.

References

1. Norman, D.: The Design of everyday things. MIT press, Cambridge, MA (1998)
2. Winters, F., Mielenze, C., Hellestrand, G.: Design Process Changes Enabling Rapid Development. In: Proc. Convergence 2004, pp. 613–624. Society of Automotive Engineers, Warrendale, PA (2004)

3. Hellestrand, G.: The Revolution in Systems Engineering. IEEE Spectrum, 43–51 (September 1999)
4. NASSDA, Maximizing Silicon ROI: The Cost of Failure and Success, Nassda White Paper WP020522-1A (2003)
5. Dix, A., Finlay, J., Abowd, G., Beal, R.: Human Machine Interaction. MacGrow-Hill, New York (2004)
6. Minin, L.: Virtual prototyping in the human-machine interaction: a benchmark analysis and tests for the selection of the virtual prototyping tool which fits with the requirements of a user interface, M.S. Thesis, University of Modena and Reggio Emilia, Italy (2005)
7. Szekely, P.: User interface prototyping: Tools and techniques (1995)
8. Myers, B.: User interface software tools. ACM Transactions on Computer-Human Interaction 2(1), 64–108 (1995)
9. Righetti, X.: Study of Prototyping Tools for User Interface Design, Thesis, University of Geneve (2006)
10. Mayhew, D.: Principles and guidelines in Software User Interface Design. Prentice Hall, Englewood Cliffs (1992)
11. Sommerville: Software Engineering, 5th edn. (1995)